In the Claims:

Please amend Claims 1, 2, 4-5, 9-11, 13-14 and 18-19; cancel Claims 6-7 and 15-16; and add new Claims 20-23, all as shown below.

1. (Currently Amended) A system including an integrated development environment for use with a mark-up language, to abstract complexity of enterprise service application program interface (API) programming, comprising:

a server computer, including a processing device and a plurality of enterprise service API for one of messaging, operation, administration, and management monitoring;

a client computer, including a processing device;

an integrated development environment that includes a graphical user interface that executes on [[a]] the client ~~machine~~ computer, and that ~~allows a user to enter a markup language program~~ receives a user input markup language program, wherein the user input markup language program specifies the name of at least one enterprise service API at the server processor, and operations to be performed therewith;

a parser that receives the user input markup language program from the integrated development environment and parses the user input markup language program to extract markup language commands;

a command processor that validates the markup language commands, and, for each markup language command converts the markup language command into a command object for communication to a command dispatcher;

a command dispatcher that receives the command objects from the command processor and for each command object, assigns the command object to one of a plurality of categories corresponding to [[a]] the ~~plurality of application program interfaces~~ enterprise service API specified in the user input markup language program;

a plurality of processor modules, including a processor module for each category of ~~application program interface~~ enterprise service API, wherein each processor module receives the command objects assigned to its category, and ~~performs appropriate operations against the corresponding application program interface~~ uses the command object to perform operations at the corresponding enterprise service API located on the server; and

wherein the integrated development environment allows the user to edit and modify the markup language program as desired to access the ~~application program interfaces~~ enterprise service API.

2. (Currently Amended) The system of claim 1 wherein the markup language is ~~Java Message Service Markup Language~~ JMSML.

3. (Previously Presented) The system of claim 1 wherein the graphical user interface includes a source editor that allows a user to enter programs as Extensible Markup Language code.

4. (Currently Amended) The system of claim 1 wherein the graphical user interface includes a design editor and a set of toolbars that allow a user to generate Extensible Markup Language source code by visually assembling commands ~~conforming to the Java Message Service specification~~ within the graphical user interface.

5. (Currently Amended) The system of claim 2 wherein the graphical user interface includes a source editor that allows a user to enter ~~Java Message Service Markup Language~~ JMSML programs as Extensible Markup Language code.

6-8. (Canceled).

9. (Currently Amended) The system of claim 1 wherein said integrated development environment is used to communicate said markup language components to ~~said remote~~ the server computer via a wide area network or the Internet.

10. (Currently Amended) A method of using an integrated development environment with a mark-up language, comprising:

~~providing an integrated development environment that includes a graphical user interface that executes on a client machine, and that allows a user to enter and edit a markup language program;~~

~~receiving the markup language program from the integrated development environment and parsing the markup language program to extract markup language commands;~~

~~validating the markup language commands, and, for each markup language command converting the markup language command into a command object for communication to a command dispatcher;~~

~~receiving command objects at the command dispatcher and, for each command object, assigning the command object to one of a plurality of categories corresponding to a plurality of application program interfaces;~~

~~communicating the command objects to a plurality of processor modules, including a processor module for each category of application program interface, wherein each processor module receives the command objects assigned to its category; and~~

~~performing appropriate operations against the corresponding application program interface, as specified by the user in the markup language program~~

providing a server computer, including a processing device and a plurality of enterprise service application program interface (API) for one of messaging, operation, administration, and management monitoring;

providing a client computer, including a client processing device;

providing an integrated development environment that includes a graphical user interface that executes on the client computer, and that receives a user input markup language program, wherein the user input markup language program specifies the name of at least one enterprise service API at the server processor, and operations to be performed therewith;

receiving the user input markup language program from the integrated development environment at a parser, and parsing the user input markup language program to extract markup language commands;

validating the markup language commands at a command processor, and, for each markup language command converting the markup language command into a command object for communication to a command dispatcher;

receiving the command objects from the command processor at a command dispatcher and, for each command object, assigning the command object to one of a plurality of categories corresponding to the enterprise service API specified in the user input markup language program;

communicating the command objects to a plurality of processor modules, including a processor module for each category of enterprise service API, wherein each processor module receives the command objects assigned to its category, and

using the command object to perform operations at the corresponding enterprise service API located on the server.

11. (Currently Amended) The method of claim 10 wherein the markup language is ~~Java Message Service Markup Language~~ JMSML.

12. (Previously Presented) The method of claim 10 wherein the graphical user interface includes a source editor that allows a user to enter programs as Extensible Markup Language code.

13. (Currently Amended) The method of claim 10 wherein the graphical user interface includes a design editor and a set of toolbars that allow a user to generate Extensible Markup Language source code by visually assembling commands ~~conforming to the Java Message Service specification~~ within the graphical user interface.

14. (Currently Amended) The method of claim 11 wherein the graphical user interface includes a source editor that allows a user to enter ~~Java Message Service Markup Language~~ JMSML programs as Extensible Markup Language code.

15-17. (Canceled)

18. (Currently Amended) The method of claim 10 wherein said integrated development environment is used to communicate said markup language components to ~~said remote~~ the server computer via a wide area network or the Internet.

19. (Currently Amended) A computer ~~readablemedium~~ program product including a storage medium having instructions stored thereon, which when executed cause the computer to perform the steps of:

~~providing an integrated development environment that includes a graphical user interface that executes on a client machine, and that allows a user to enter and edit a markup language program;~~

~~receiving the markup language program from the integrated development environment and parsing the markup language program to extract markup language commands;~~

~~validating the markup language commands, and, for each markup language command converting the markup language command into a command object for communication to a command dispatcher;~~

~~receiving command objects at the command dispatcher and, for each command object, assigning the command object to one of a plurality of categories corresponding to a plurality of application program interfaces;~~

~~communicating the command objects to a plurality of processor modules, including a processor module for each category of application program interface, wherein each processor module receives the command objects assigned to its category; and~~

~~performing appropriate operations against the corresponding application program interface, as specified by the user in the markup language program~~

providing a server computer, including a processing device and a plurality of enterprise service application program interface (API) for one of messaging, operation, administration, and management monitoring;

providing a client computer, including a client processing device;

providing an integrated development environment that includes a graphical user interface that executes on the client computer, and that receives a user input markup language program,

wherein the user input markup language program specifies the name of at least one enterprise service API at the server processor, and operations to be performed therewith;

receiving the user input markup language program from the integrated development environment at a parser, and parsing the user input markup language program to extract markup language commands;

validating the markup language commands at a command processor, and, for each markup language command converting the markup language command into a command object for communication to a command dispatcher;

receiving the command objects from the command processor at a command dispatcher and, for each command object, assigning the command object to one of a plurality of categories corresponding to the enterprise service API specified in the user input markup language program;

communicating the command objects to a plurality of processor modules, including a processor module for each category of enterprise service API, wherein each processor module receives the command objects assigned to its category, and

using the command object to perform operations at the corresponding enterprise service API located on the server.

20. (New) The system of claim 1 wherein the parser, command processor, command dispatcher and plurality of processor modules are located on the client computer.

21. (New) The system of claim 1 wherein the parser, command processor, command dispatcher and plurality of processor modules are located on the server computer.

22. (New) The method of claim 10 wherein the parser, command processor, command dispatcher and plurality of processor modules are located on the client computer.

23. (New) The method of claim 10 wherein the parser, command processor, command dispatcher and plurality of processor modules are located on the server computer.